Evaluating biological plausibility of learning algorithms the lazy way

Anonymous Author(s)
Affiliation
Address
email

Abstract

To which extent can successful machine learning inform our understanding of biological learning? One popular avenue of inquiry in recent years has been to directly map such algorithms into a realistic circuit implementation. Here we focus on learning in recurrent networks and investigate a range of learning algorithms. Our approach decomposes them into their computational building blocks and discusses their abstract potential as biological operations. This alternative strategy provides a "lazy" but principled way of evaluating ML ideas in terms of their biological plausibility.

The phenomenal success of neurally inspired machine learning (ML) algorithms has captured the imagination of many neuroscientists hoping to understand the principles of learning in the brain. This led to renewed efforts to find biologically realistic approximations to the backpropagation algorithm, the key to efficient credit assignment in feedforward networks (1; 2; 3; 4; 5). Relatively little work covers the—arguably more realistic—scenario of learning in recurrent networks and *temporal* credit assignment (6). The canonical machine learning solution, **backpropagation through time** (BPTT, (7)), poses more severe challenges when attempting to map it to biology. In particular, BPTT is *temporally nonlocal*; that is, each weight update depends on network activity across multiple time points. This problem could be addressed by using **online** alternatives to BPTT such as Real-Time Recurrent Learning (RTRL, (8)). Unfortunately, RTRL has an $\mathcal{O}(n^3)$ memory requirement (for n neurons) and it is hard to imagine any biological structure for storing this many real-valued variables. The cubic complexity is a problem not only for biology, but also for machine learning and has precluded the use of RTRL in applications. Nonetheless, the practical need for efficient online learning has revived the RTRL idea, leading to several new algorithms (9; 10; 11; 12; 13), which all reduce memory complexity to $\mathcal{O}(n^2)$, but differ in the nature of their approximations (14). Our goal here is to investigate to what extent these ideas could be implemented in a biological circuit.

One *could* take each algorithm individually and try to model in detail a biophysical implementation, à la (1; 2; 3; 4; 5). However, it's unlikely that any single ML solution maps one-to-one onto neural circuitry. Instead, a more useful exercise would be to identify core computational building blocks that are strictly necessary for solving temporal credit assignment, which are more likely to have a direct biological analogue. To this end, we put forward a principled framework for evaluating biological plausibility in terms of the mathematical operations required–hence our "lazy" analysis. We examine several online algorithms within this framework, identifying potential issues common across algorithms, for example the need to physically represent the Jacobian of the network dynamics. We propose some novel solutions to this and other issues and in the process articulate biological mechanisms that could facilitate these solutions. Finally, we empirically validate that these biologically realistic approximations still solve temporal credit assignment, in two simple synthetic tasks.

Plausibility criteria for recurrent learning. Consider a recurrent network of n units, with voltages $\mathbf{v}^{(t)} = \mathbf{W}\hat{\mathbf{r}}^{(t-1)}$, where $\hat{r}^{(t)}$ is the concatenation of recurrent and external inputs, with an additional constant input for the bias term, $\hat{\mathbf{r}}^{(t-1)} = [\mathbf{r}^{(t-1)}; \mathbf{x}^{(t)}; 1] \in \mathbb{R}^m$ ($m = n + n_{\text{in}} + 1$) and trainable weights organized as $\mathbf{W} = [\mathbf{W}^{\text{rec}}, \mathbf{W}^{\text{in}}, \mathbf{b}^{\text{rec}}] \in \mathbb{R}^{n\times m}$. For a closer match to neural circuits, the firing rates update in *continuous* time, via $\mathbf{r}^{(t)} = (1-\alpha)\mathbf{r}^{(t-1)} + \alpha\phi(\mathbf{v}^{(t)})$, using a point-wise neural activation function $\phi : \mathbb{R}^n \to \mathbb{R}^n$ (e.g. $\tanh$) and the network's inverse time constant $\alpha \in (0, 1]$. The network output $\mathbf{y}^{(t)} = \text{softmax}(\mathbf{W}^{\text{out}}\mathbf{r}^{(t)} + \mathbf{b}^{\text{out}}) \in \mathbb{R}^{n_{\text{out}}}$ is computed by output weights/bias $\mathbf{W}^{\text{out}} \in \mathbb{R}^{n_{\text{out}}\times n}$, $\mathbf{b}^{\text{out}} \in \mathbb{R}^{n_{\text{out}}}$ and compared with the training label $\mathbf{y}^{*(t)}$ to produce an instantaneous loss $L^{(t)}$.

BPTT and RTRL each provide a method for calculating the gradient of each instantaneous loss $\partial L^{(t)}/\partial W_{ij}$, to be used for gradient descent. BPTT unrolls the network over time and performs backpropagation as if on a feedforward network:

$$\frac{\partial L^{(t)}}{\partial W_{ij}} = \sum_{t'\leq t} \left(\bar{\mathbf{c}}^{(t)} \prod_{s=t'+1}^{t} \mathbf{J}^{(s)} \right)_i \alpha\phi'(v_i^{(t')})\hat{r}_j^{(t'-1)}, \tag{1}$$

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

Name	Tensor(s)	Update equations	Notes
UORO	$A_k^{(t)} B_{ij}^{(t)}$	$A_k^{(t)} = \rho_0 \sum_{k'} J_{kk'}^{(t)} A_{k'}^{(t-1)} + \rho_1 \nu_k$	$\boldsymbol{\nu}$ i.i.d. $\sim \{\pm 1\}$, $\rho_0, \rho_1 > 1$ constants that can
		$B_{ij}^{(t)} = \rho_0^{-1} B_{ij}^{(t-1)} + \rho_1^{-1} \nu_i \phi'(v_i^{(t)}) \hat{r}_j^{(t-1)}$	be chosen to minimize variance
KF-RTRL	$A_j^{(t)} B_{ki}^{(t)}$	$A_j^{(t)} = \nu_0 \rho_0 A_j^{(t-1)} + \nu_1 \rho_1 \hat{r}_j^{(t-1)}$	
		$B_{ki}^{(t)} = \nu_0 \rho_0^{-1} \sum_{k'} J_{kk'}^{(t)} B_{k'i}^{(t-1)} + \nu_1 \rho_1^{-1} \alpha \delta_{ki} \phi'(v_i^{(t)})$	
R-KF	$A_i^{(t)} B_{kj}^{(t)}$	$A_i^{(t)} = \rho_0 A_i^{(t-1)} + \rho_1 \nu_i$	
		$B_{kj}^{(t)} = \rho_0^{-1} \sum_{k'} J_{kk'}^{(t)} B_{k'j}^{(t-1)} + \rho_1^{-1} \nu_i \phi'(v_i^{(t)}) \hat{r}_j^{(t-1)}$	
KeRNL	$A_{ki}^{(t)} B_{ij}^{(t)}$	$A_{ki}^{(t)} = A_{ki}^{(t-1)} - \eta_A \left(\sum_{i'} A_{ki'}^{(t-1)} \zeta_{i'}^{(t)} - \Delta r_k^{(t)} \right) \zeta_i^{(t)}$	$\Delta r_k^{(t)}$ noise effects, $\zeta_i^{(t)}$ filtered noise
		$B_{ij}^{(t)} = (1 - \alpha_i) B_{ij}^{(t-1)} + \alpha_i \phi'(v_i^{(t)}) \hat{r}_j^{(t-1)}$	η_A learning rate, α_i learned timescales
RFLO	$\delta_{ki}^{(t)} B_{ij}^{(t)}$	$B_{ij}^{(t)} = (1 - \alpha) B_{ij}^{(t-1)} + \alpha \phi'(v_i^{(t)}) \hat{r}_j^{(t-1)}$	α timescale of forward pass
DNI	$A_{li}^{(t)}$	$A_{li}^{(t)} = A_{li}^{(t-1)} - \eta_A \left[\sum_{l'} \bar{r}_{l'}^{(t)} A_{l'i}^{(t-1)} - c_i^{*(t)} \right]$	l indexes entries of $\bar{\mathbf{r}}^{(t)} = [\mathbf{r}^{(t)}; \mathbf{y}^{*(t)}; 1]$.
		where $c_i^{*(t)} = \bar{c}_i^{(t)} + \sum_m \sum_{l'} \bar{r}_{l'}^{(t+1)} A_{l'm}^{(t-1)} J_{mi}^{(t+1)}$	

Table 1: A summary of several new online algorithms' tensor structure and update equations.

where $\bar{\mathbf{c}}^{(t)} \equiv \partial L^{(t)} / \partial \mathbf{r}^{(t)} \in \mathbb{R}^n$ is the immediate credit assignment vector and $\mathbf{J}^{(s)} \equiv \partial \mathbf{r}^{(s)} / \partial \mathbf{r}^{(s-1)}$ is the network Jacobian, with elements $J_{ij}^{(s)} = (1 - \alpha)\delta_{ij} + \alpha \phi'(h_i^{(s)}) W_{ij}^{\text{rec}}$. While Eq. (1) explicitly references activity at all time points, RTRL instead recursively updates the "influence tensor" $M_{kij}^{(t)} = \partial r_k^{(t)} / \partial W_{ij}$ by $M_{kij}^{(t)} = \sum_{k'} J_{kk'}^{(t)} M_{k'ij}^{(t-1)} + \alpha \delta_{ki} \phi'(v_i^{(t)}) \hat{r}_j^{(t-1)}$, preserving the first-order long-term dependencies in the network as it runs forward. The actual gradient is then calculated as

$$\frac{\partial L^{(t)}}{\partial W_{ij}} = \sum_k \frac{\partial L^{(t)}}{\partial r_k^{(t)}} \frac{\partial r_k^{(t)}}{\partial W_{ij}} = \sum_k \bar{c}_k^{(t)} M_{kij}^{(t)}. \tag{2}$$

Unlike BPTT, every computation in RTRL involves only current time t or $t-1$. In general, an online algorithm has some **tensor structure** for summarizing the inter-temporal dependencies in the network, to avoid having to explicitly unroll the network. These tensor(s) must **update** at each time step as new data come in. RTRL uses an order-3 tensor, resulting in an $\mathcal{O}(n^3)$ memory requirement that is neither efficient nor biologically plausible. However, all of the new online algorithms we discuss are only $\mathcal{O}(n^2)$ in memory. In Table 1 we show the tensor structure and update equations for each of these algorithms in order to discuss the mathematical operations needed for each and whether a neural circuit could implement them. How these updates lead to sensible learning is outside our scope, and we refer the reader to either the original papers (9; 10; 12; 11; 13) or the review (14).

In a purely artificial setting, these tensor updates from Table 1 are straightforward to implement, but biologically, one has to consider how these tensors are physically represented and the mechanism for performing the updates. We present a list of mathematical operations and comment on how a biological neural network might or might not be able to perform it in parallel with the forward pass:

- i A vector can be encoded as a firing rate, voltage, or any other intracellular variable.
- ii A matrix must be encoded as the strengths of a set of synapses; if individual entries change, they must do so time-continuously and via a (local) synaptic plasticity rule.
- iii Matrix-vector multiplication can be implemented by neural transmission, but input vectors must represent firing rates, as opposed to voltages or other intracellular variables.
- iv Matrix-matrix multiplication is at face value not possible, as it requires $\mathcal{O}(n^3)$ multiplications, and there is no biological structure to support this.
- v Independent additive noise is feasible; biological neural networks are naturally noisy in ways that can be leveraged for computation.
- vi At face value, it is not possible to maintain a "noisy" copy of the network to estimate perturbation effects, e.g. KeRNL (Table 1) or (15). However, there may be workarounds.

How do different algorithms do? RFLO is sufficiently simple to pass all of these tests, but it arguably doesn't actually solve temporal credit assignment and merely regresses natural memory traces to task labels (see Section 5.5 of (14)), which limits its performance ceiling. Every other algorithm fails at least one of these criteria, at least at first glance. KF-RTRL and R-KF are out because of the matrix-matrix products in their updates. Although the eligibility-trace-like update in KeRNL for $B_{ij}^{(t)}$ is straightforward, learning the $A_{ki}^{(t)}$ matrix requires a perturbed network–on the surface unlikely biologically (vi). While UORO uses only matrix-vector products, the time-continuity requirement (ii) is awkward, because if we choose the constants ρ_0, ρ_1 to make one update equation smooth in time (e.g. $\rho_0 = 1 - \epsilon, \rho_1 = \epsilon$, for $0 < \epsilon \ll 1$), the other update becomes unstable due to the appearance of ρ_0^{-1}, ρ_1^{-1}. DNI avoids matrix-matrix

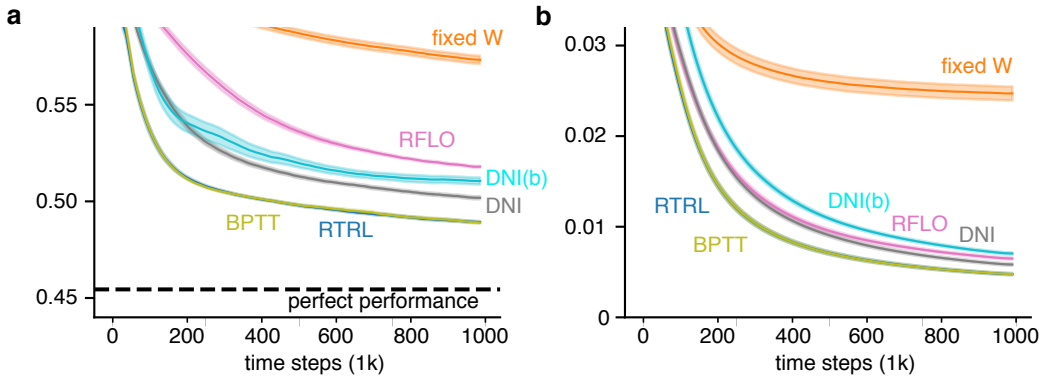

Figure 1: **a)** Cross-entropy loss for networks trained on Add task with $\alpha = 0.5$, $t_1 = 3$, and $t_2 = 5$ for various algorithms. Lines are means over 20 random seeds (weight init. and training examples), and shaded regions represent ± 1 S.E.M. Raw loss curves are first down-sampled by a factor of 10^{-4} (rect. kernel) and then smoothed with a 10-time-step running average. **b)** Same for mean-squared error on Mimic task.

products and is naturally time-continuous, but it involves a matrix-matrix-vector product, requiring physical storage of the intermediate product. To top it off, every algorithm (except RFLO) requires the Jacobian of the network, but superficially this is not possible (ii), because a set of synapses cannot instantaneously readjust to their ideal values $J_{ij}^{(t)} = (1-\alpha)\delta_{ij} + \alpha\phi'(v_i^{(t)})\hat{r}_j^{(t-1)}$.

Can we fix any of these issues? While each algorithm poses its own challenges, the Jacobian is a recurring problem for anything that meaningfully solves credit assignment. Therefore we propose a general solution, to instead use an *approximate* Jacobian, whose entries we refer to as $\mathcal{J}_{ij}$, which updates at each step according to a perceptron-like learning rule:

$$\Delta\mathcal{J}_{ij} \propto -(r_i^{(t)} - \sum_{j'} \mathcal{J}_{ij'} r_{j'}^{(t-1)}) r_j^{(t-1)}. \tag{3}$$

Biologically, this would correspond to having an additional set of synapses (possibly spatially segregated from $\mathbf{W}$) with their own plasticity rules (16). Computationally, this approximation brings no traditional speed benefits, but it offers a plausible mechanism by which a neural circuit can access its own Jacobian for learning purposes.

As for other challenges, the matrix-matrix-vector product appearing in DNI can be implemented by the circuit itself in a phase of computation separate from the forward pass. For the intermediate result to pass through the second matrix, it must be represented as a firing rate (iii), which already requires altering the original equations to $\sum_m \phi\left(\sum_{l'} \bar{r}_{l'}^{(t+1)} A_{l'm}\right) J_{mi}$, since on its own $u_m^{(t+1)} \equiv \sum_{l'} \bar{r}_{l'}^{(t+1)} A_{l'm}$ is a voltage. This would naively interfere with the forward pass, since $\mathbf{v}^{(t)} = \mathbf{W}\hat{\mathbf{r}}^{(t-1)}$ already uses the network firing rates and somatic voltages. However, we could imagine the $\mathbf{A}$ synapses feeding into an electrically isolated neural compartment (say the apical dendrites) to define a separate voltage $u_m^{(t+1)}$, which is allowed to drive neural firing to $\phi(u_m^{(t+1)})$ in specific "update" phases. We already know that branch-specific gating (by interneurons) can filter which information makes it to the soma to drive spiking (17).

Do these fixes work empirically? Given our criteria and novel workarounds, RFLO and DNI(b), our altered version of DNI (which includes the approximate Jacobian), remain as viable candidates for neural learning. To ensure our additional approximations do not ruin performance, we empirically evaluate DNI(b), along with the original DNI and RFLO. As upper and lower bounds on performance, respectively, we also include exact credit assignment methods (BPTT and RTRL) and a "fixed-$\mathbf{W}$" algorithm that only trains the output weights. We use two synthetic tasks, each of which requires solving temporal credit assignment and has clear markers for success. One task ("Add") requires mapping a stream of i.i.d. Bernoulli inputs $x^{(t)}$ to an output $y^{*(t)} = 0.5 + 0.5x^{(t-t_1)} - 0.25x^{(t-t_2)}$ (18), with time rescaled to match α. The label depends on the inputs via lags t_1, t_2 that can be adjusted to modulate task difficulty. The other task ("Mimic") requires reproducing the response of a separate RNN with the same architecture and fixed weights to a shared Bernoulli input stream. We find that training loss for RFLO and DNI is worse than the optimal solutions (BPTT and RTRL), but both beat the fixed-$\mathbf{W}$ performance lower bound. DNI(b) performs worse than original DNI, unsurprising because it involves further approximations, but still much better than the fixed-$\mathbf{W}$ baseline. This demonstrates that solving temporal credit assignment is possible within biological constraints.

Discussion

It is still unclear how neural circuits achieve sophisticated learning, in particular solving temporal credit assignment. Here we approached the problem by looking for biologically sensible approximations to RTRL and BPTT. Although we have empirical results to prove that our solutions can solve temporal credit assignment for simple tasks, the substance of our contribution is conceptual, in articulating what computations are abstractly feasible and which are not. In particular, we have shown that accessing the Jacobian for learning is possible by using a set of synapses trained to linearly approximate the network's own dynamics.

Along the way, we have identified some key lessons. The main one is that neural circuits need additional infrastructure specifically to support learning. This could be extra neurons, extra compartments within neurons, separate coordinated phases of computation, input

gating by inhibition, etc. While we all know that biology is a lot more complicated than traditional models of circuit learning would suggest, it has proved difficult to identify the functional role of these details in a bottom-up way. On the other hand, drawing a link between ML algorithms and biology can hint at precise computational roles for not well understood circuit features.

Another lesson is that implementing even fairly simple equations in parallel to the forward pass is nontrivial, since it already uses up so much neural hardware. Even a simple matrix-vector product requires an entirely separate phase of network dynamics in order to not interfere with the forward pass of computation. While it may be tempting to outsource some of these update equations to separate neurons, the results would not be locally available to drive synaptic plasticity.

Of course, we acknowledge that any particular solution, whether RFLO or DNI, is a highly contrived, specific, and likely incorrect guess at how neural circuits learn, but we believe the exercise has big-picture implications for how to think about biological learning. Beyond the particular topic of online learning in recurrent networks, our work provides a general blueprint for abstractly evaluating computational models as mechanistic explanations for biological neural networks. Knowing what computational building blocks are at our disposal and what biological details are needed to implement them is an important foundation for studying ML algorithms in a biological context.

References

[1] J. Sacramento, R. P. Costa, Y. Bengio, and W. Senn, "Dendritic cortical microcircuits approximate the backpropagation algorithm," in *Advances in Neural Information Processing Systems*, pp. 8721–8732, 2018.

[2] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, p. 13276, 2016.

[3] J. Guerguiev, T. P. Lillicrap, and B. A. Richards, "Towards deep learning with segregated dendrites," *ELife*, vol. 6, p. e22901, 2017.

[4] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[5] B. Scellier and Y. Bengio, "Equilibrium propagation: Bridging the gap between energy-based models and backpropagation," *Frontiers in computational neuroscience*, vol. 11, p. 24, 2017.

[6] T. P. Lillicrap and A. Santoro, "Backpropagation through time and the brain," *Current opinion in neurobiology*, vol. 55, pp. 82–89, 2019.

[7] P. J. Werbos *et al.*, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[8] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[9] C. Tallec and Y. Ollivier, "Unbiased online recurrent optimization," *arXiv preprint arXiv:1702.05043*, 2017.

[10] A. Mujika, F. Meier, and A. Steger, "Approximating real-time recurrent learning with random kronecker factors," in *Advances in Neural Information Processing Systems*, pp. 6594–6603, 2018.

[11] C. Roth, I. Kanitscheider, and I. Fiete, "Kernel rnn learning (kernl)," 2018.

[12] J. M. Murray, "Local online learning in recurrent networks with random feedback," *eLife*, vol. 8, p. e43299, 2019.

[13] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1627–1635, JMLR. org, 2017.

[14] O. Marschall, K. Cho, and C. Savin, "A unified framework of online learning algorithms for training recurrent neural networks," *arXiv preprint arXiv:1907.02649*, 2019.

[15] I. R. Fiete and H. S. Seung, "Neural network models of birdsong production, learning, and coding," *New Encyclopedia of Neuroscience. Eds. L. Squire, T. Albright, F. Bloom, F. Gage, and N. Spitzer. Elsevier*, 2007.

[16] U. Gordon, A. Polsky, and J. Schiller, "Plasticity compartments in basal dendrites of neocortical pyramidal neurons," *Journal of Neuroscience*, vol. 26, no. 49, pp. 12717–12726, 2006.

[17] P. Somogyi, L. Katona, T. Klausberger, B. Lasztóczi, and T. J. Viney, "Temporal redistribution of inhibition over neuronal subcellular domains underlies state-dependent rhythmic change of excitability in the hippocampus," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 369, no. 1635, p. 20120518, 2014.

[18] S. Pitis, "Recurrent neural networks in tensorflow 1." `r2rt.com/recurrent-neural-networks-in-tensorflow-i`, 2016. Accessed: 2018-11-13.